Pembelajaran Dalam (Deep Learning)

Chapter 2
Multilayers Artificial Neural Networks (ANN)

Genap TA 2022/2023

FYS, ITQ | Maret 2023

Multilayer ANN

1. Pendahuluan Pembelajaran Mesin dan Pembelajaran Dalam (CLO 1)
2. Programming Python: Dasar Numpy, Pandas, Matpotlib, dll (CLO 3)
3. Dasar Regresi dan Klasifikasi (CLO 1)
4. Artificial Neural Networks (ANN) (CLO 2/3)
5. **Multi-Layer Perceptron (MLP) (CLO 2/3)**
6. TensorFlow (CLO 3)
7. TensorFlow (+Kinerja Machine Learning) (CLO 3/1)
8. TensorFlow Lanjutan (CLO 3)
9. Convolutional Neural Networks (CNN) (CLO 2/3)
10. CNN Lanjutan (CLO 2/3)
11. Recurrent Neural Networks (RNN) (CLO 2/3)
12. RNN Lanjutan (CLO 2/3)
13. Generative Adversarial Networks (GANs) (CLO 2/3)
14. GANs lanjutan (CLO 2/3)
15. Aplikasi Object Detections (CLO 2/3)
16. UAS (CLO 2/3)

Multilayer ANN

Course Learning Outcomes (CLO)

- CLO 2: Mahasiswa dapat mendeskripsikan ide dasar dan cara kerja algoritma pembelajaran dalam (deep learning)
- CLO 3: Mahasiswa dapat mengimplementasikan algoritma pembelajaran dalam yang populer menggunakan perangkat lunak

Multilayer ANN

Tujuan Perkuliahan

- Memahami konsep **Multilayer Artificial Neural Networks**
- Memahami dan mengimplementasikan algoritma **Backpropagation** untuk men-training ANN dari awal (*from scratch*)
- Men-training multilayers neural networks untuk klasifikasi image

Multilayer ANN

Outline:

A. Memodelkan Fungsi Kompleks dengan ANN

- Neural Networks dengan Satu Layer (recap)
- Pengenalan Arsitektur Multilayer Neural Networks
- Aktivasi Neural Networks dengan Forward Propagation

B. Training Artificial Neural Networks

- Menghitung Cost Function Logistik
- Intuisi untuk Algoritma Backpropagation
- Training Neural Networks via Backpropagation

Multilayer ANN

- Pada pertemuan sebelumnya dapat diambil kesimpulan dalam Machine Learning secara umum mempunyai langkah:
  1. Melihat pola
  2. Memodelkan pola dengan persamaan matematis
  3. Mencari parameter model dengan objective function atau cost function yang diminimumkan (proses training/learning)
  4. Implementasi menggunakan model dengan parameter optimum yang telah diketahui melalui proses training/learning

Bagian ini akan menjelaskan langkah 2 dan 3 untuk Multilayers Artificial Neural Networks

Multilayer ANN

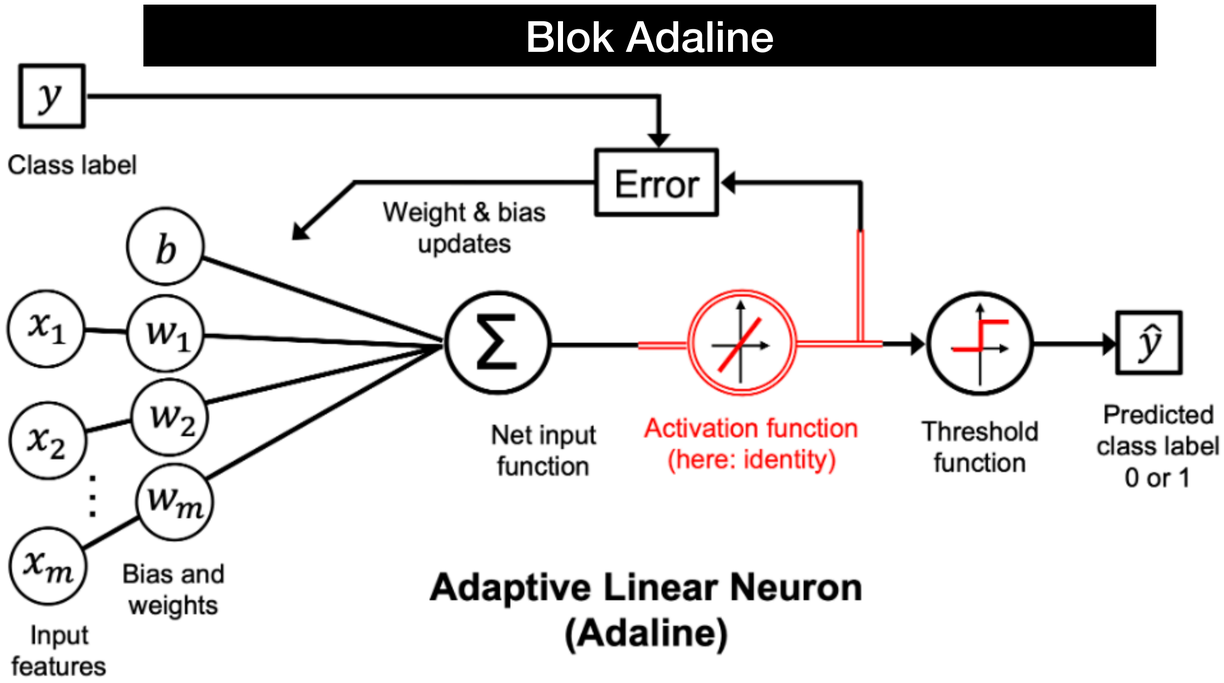
A. Memodelkan Fungsi Kompleks dengan ANN

Bagaimana mendefinisikan secara formal (memodelkan) Multilayer Artificial Neural Networks?

Multilayer ANN

A.1. Neural Networks dengan Satu Layer (Recap)

- Contoh neural networks dengan 1 layer, yaitu Adaline, telah dijelaskan di Bab 1



- Tiap epoch, update pembobot

$$\mathrm{w} = \mathrm{w} + \Delta w$$

dimana $\Delta w = -\eta \nabla J(\mathrm{w})$

- $\nabla J(\mathrm{w})$ adalah turunan parsial terhadap masing-masing bobot w_j

$$\frac{\partial J(\mathrm{w})}{\partial w_j} = -\sum_i \left(y^{(i)} - a^{(i)}\right) x_j^{(i)}$$

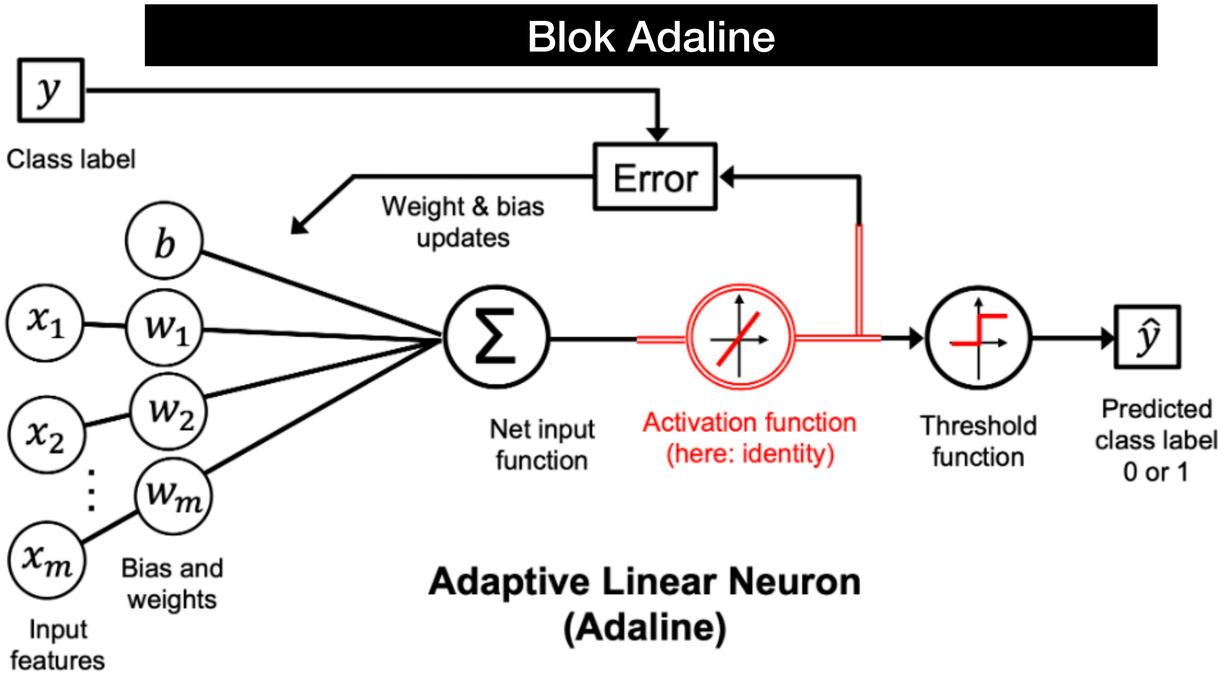
$y^{(i)}$: class target label dari $x^{(i)}$

$a^{(i)}$: fungsi aktivasi neuron

Multilayer ANN

A.1. Neural Networks dengan Satu Layer (Recap)

- Contoh neural networks dengan 1 layer, yaitu Adaline, telah dijelaskan di Bab 1



- Fungsi aktivasi $\phi(z)$ didefinisikan:

$$\phi(z) = z = a$$

- Net input z adalah kombinasi linier dari pembobot yang meaghubungkan input dan output

$$z = \sum_j w_j x_j = \mathbf{w}^T \mathbf{x}$$

- Fungsi threshold untuk mengubah output kontinyu menjadi predicted class label $\hat{y}$:

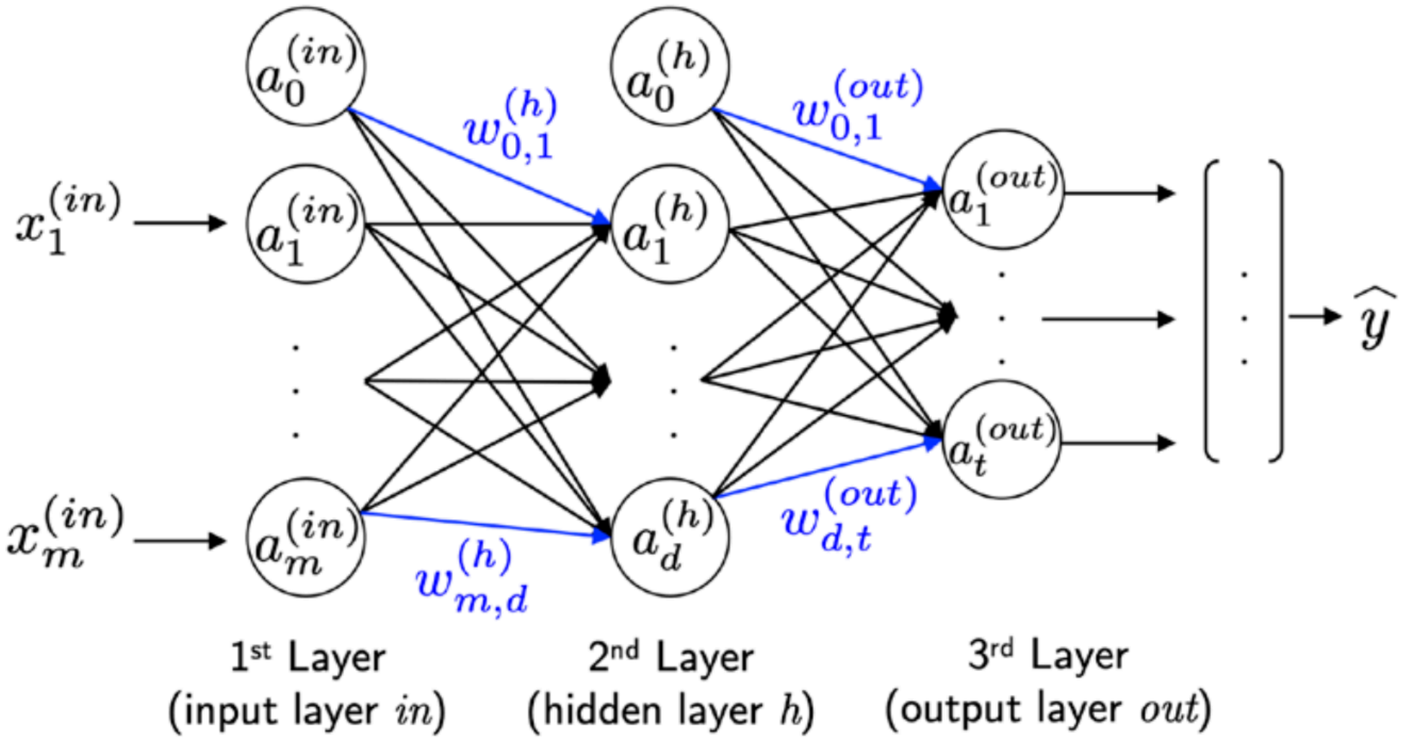
$$\hat{y} = \begin{cases} 1 & \text{if } g(z) \geq 0; \\ -1 & \text{otherwise} \end{cases}$$

Multilayer ANN

A.2. Pengenalan Arsitektur Multilayer Neural Networks

- *Multilayer feedforward neural networks* terdiri dari koneksi neuron-neuron tunggal (*single layer neuron*).
- **Multilayer Perceptron** (MLP) adalah tipe khusus dari **fully connected neural networks**
- MLP 3-layer mempunyai satu layer input, satu layer output dan satu *hidden layer*
- *Hidden layer* terkoneksi seluruhnya ke layer input dan layer input terkoneksi seluruhnya ke *hidden layer*

MLP dengan tiga layer



Network dengan lebih dari satu hidden layer disebut dengan Deep Neural Networks (DNN)

Multilayer ANN

A.2. Pengenalan Arsitektur Multilayer Neural Networks

- Hidden layer pada MLP dapat ditambah lebih banyak untuk menghasilkan arsitektur network lebih dalam
- Sejumlah layer dan unit-unit pada neural network dapat dijadikan **hyperparameter** tambahan yang akan dioptimasi
- Semakin banyak layer, maka gradient-gradient error yang akan dihitung dengan algoritma backpropagation akan semakin mengecil, disebut dengan **vanishing gradient** yang membuat learning model lebih tenanting
- Algoritma-algoritma yang khusus dibangun untuk menolong training seperti struktur DNN, disebut dengan **Deep Learning (Pembelajaran Dalam)**

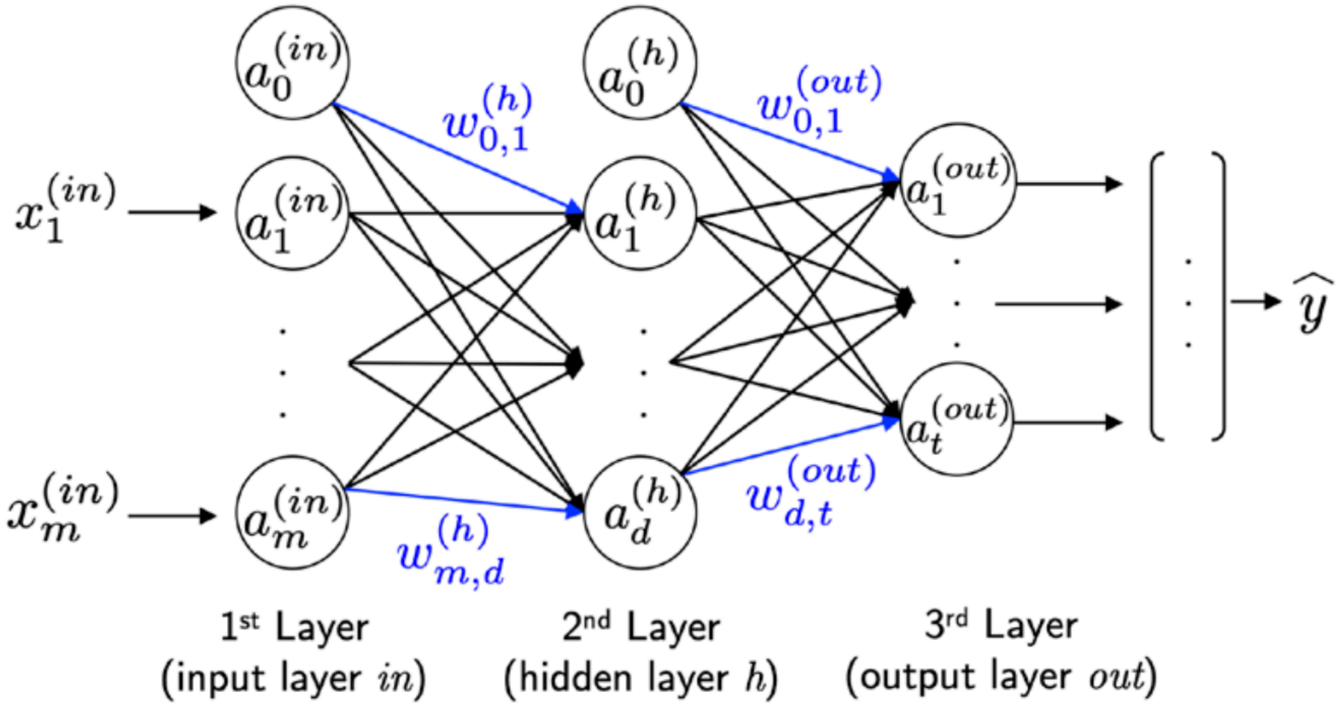
Multilayer ANN

A.2. Pengenalan Arsitektur Multilayer Neural Networks

- $a_i^{(l)}$: unit aktivasi ke-i pada layer ke-l
- Untuk layer akan digunakan superscript:
  - in : layer input
  - h : hidden layer
  - out : layer output
- Aktivasi pada layer input hanya input-nya itu sendiri ditambah uit bias:

$$\mathbf{a}^{(in)} = \begin{bmatrix} a_0^{(in)} \\ a_1^{(in)} \\ \vdots \\ a_m^{(in)} \end{bmatrix} = \begin{bmatrix} 1 \\ x_1^{(in)} \\ \vdots \\ x_m^{(in)} \end{bmatrix}$$

MLP dengan tiga layer

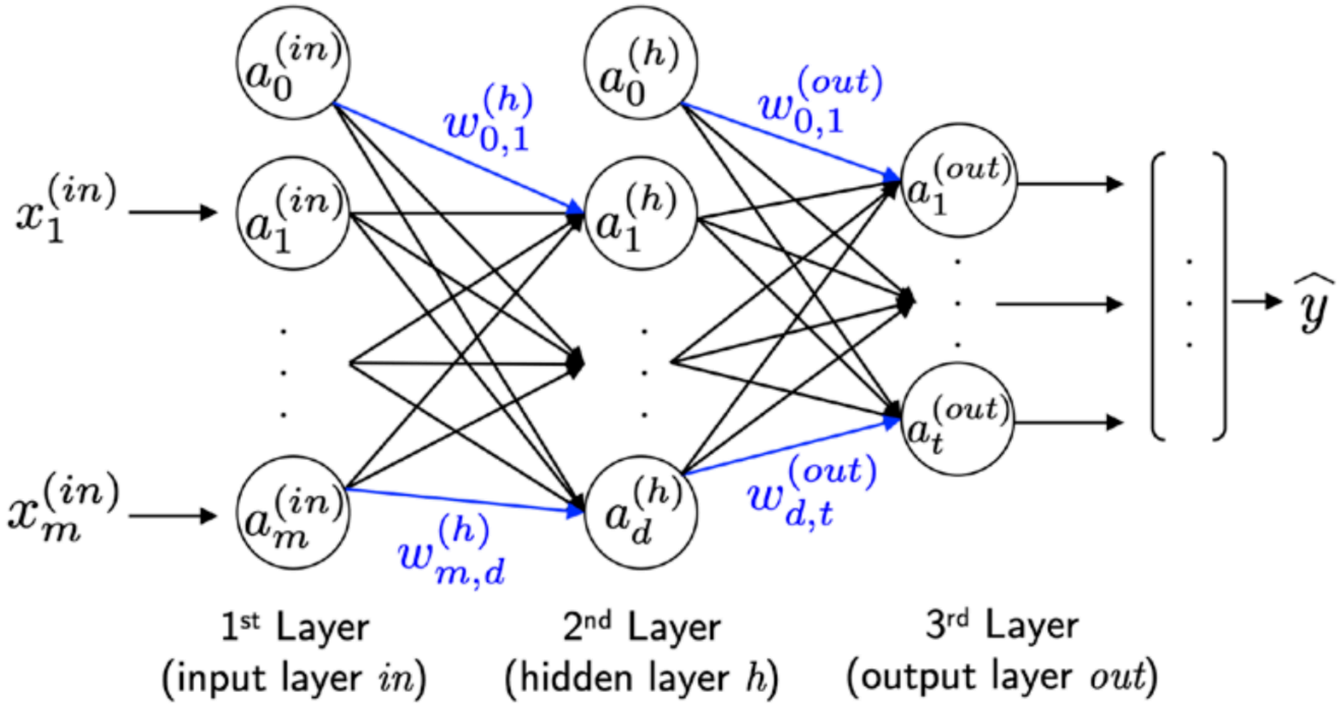


Multilayer ANN

A.2. Pengenalan Arsitektur Multilayer Neural Networks

- Setiap unit pada layer ke-l terkoneksi dengan semua unit-unit pada layer ke-$l+1$ melalui koefisien-koefisien pembobot
- $w_{k,j}^{(l)}$ adalah koefisien pembobot yang menghubungkan unit ke-k pada layer l dan unit ke-j pada layer ke-$l+1$
- Notasi matriks pembobot:
  - $\mathbf{W}^{(h)}$: menghubungkan input ke *hidden layer*
  - $\mathbf{W}^{(out)}$: menghubungkan *hidden layer* ke layer output

MLP dengan tiga layer



Multilayer ANN

A.2. Pengenalan Arsitektur Multilayer Neural Networks

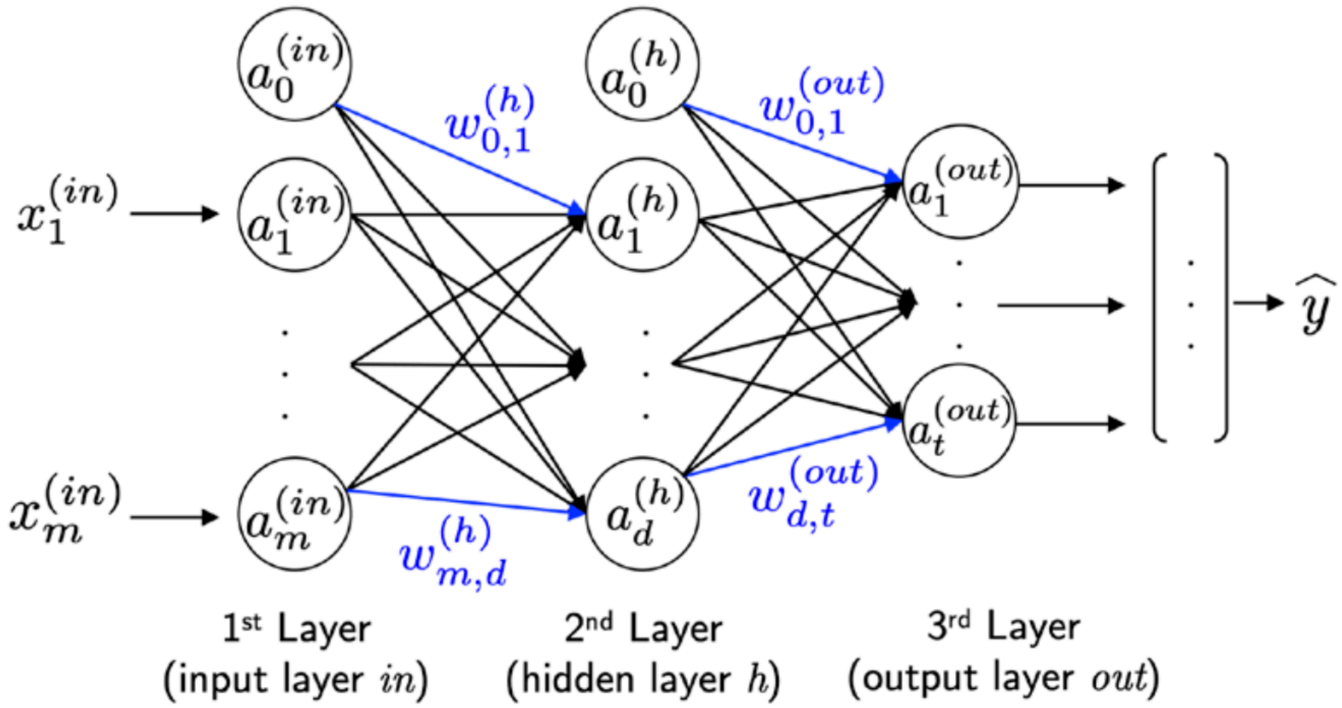
- Pada klasifikasi multiclass kita gunakan **one-hot representation** untuk variabel kategorial

- Contoh: 3-class pada dataset bunga Iris (0 = Sentosa, 1 = Versicolor, 3 = Virginica) direpresentasikan dengan

$$0 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad 1 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \quad 2 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

- Dengan *one-hot vector* ini, persoalan klasifikasi dengan jumlah label class sembarang bisa diselesaikan

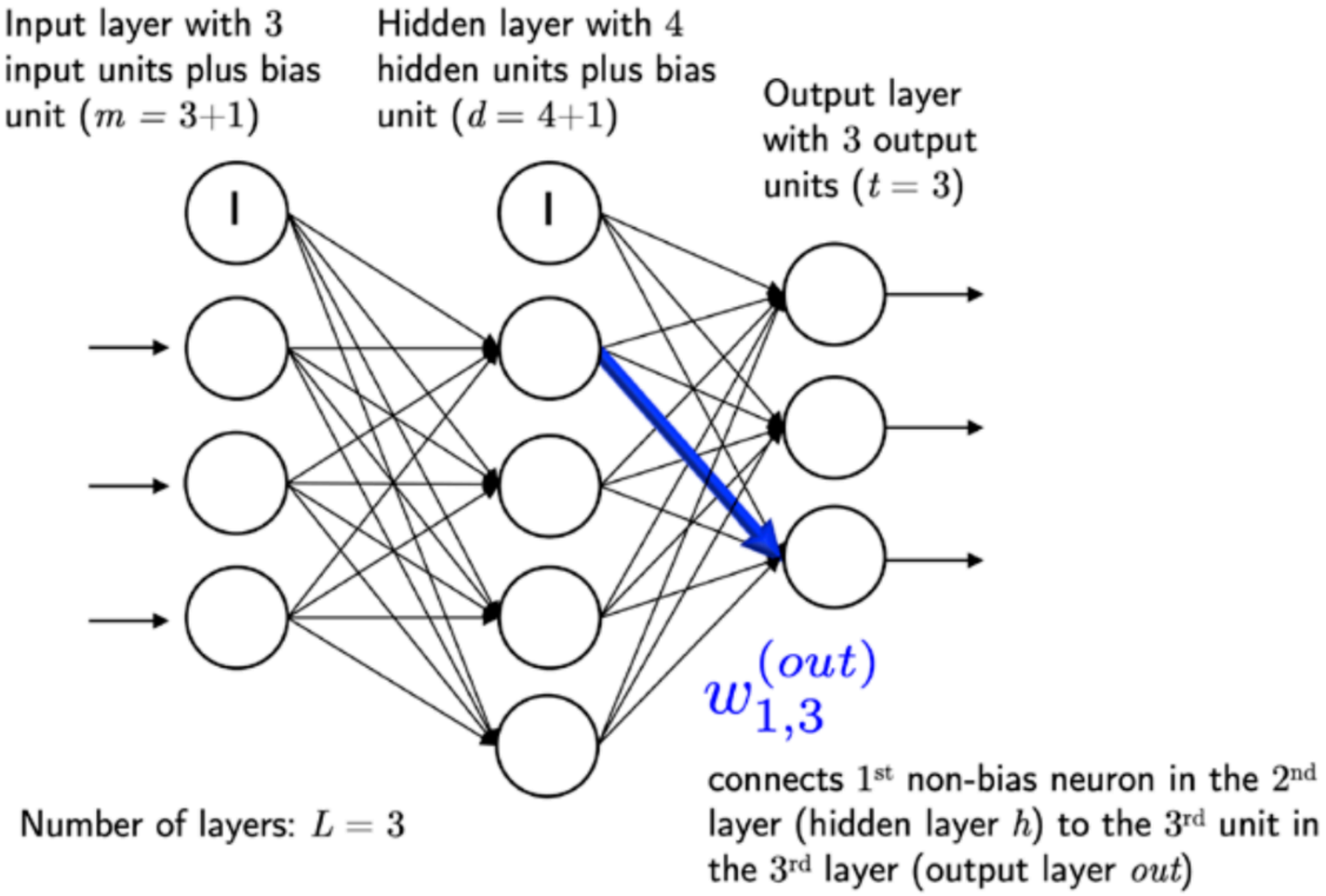
MLP dengan tiga layer



Multilayer ANN

A.2. Pengenalan Arsitektur Multilayer Neural Networks

MLP $3-4-3$



- Contoh MLP $3-4-3$
  - $\mathbf{W}^{(h)} \in \mathbb{R}^{m \times d}$: menghubungkan input dengan jumlah unit $m = 3+1$ ke *hidden layer* dengan jumlah unit $d = 4+1$
  - $\mathbf{W}^{(out)} \in \mathbb{R}^{d \times t}$: menghubungkan *hidden layer* dengan jumlah unit $d = 4+1$ ke layer output dengan jumlah unit output $t = 3$

Multilayer ANN

A.3. Aktivasi Neural Network dengan *Forward Propagation*

- Prosedur learning pada MLP secara singkat:
  1. Mulai pada layer input, pola data training diteruskan melalui network untuk menghasilkan output (**forward propagation**)
  2. Berdasarkan output network, error yang akan diminimalkan menggunakan cost function kemudian dihitung
  3. Error dipropagasi balik (*backpropagate*), turunannya terhadap masing-masing bobot pada network dihitung, kemudian mengupdate model
- Ketiga langkah diulang untuk banyak *time epoch* dan *learning* bobot-bobot pada MLP
- Menggunakan *forward propagation* kembali untuk menghitung output dan mengimplementasikan fungsi *threshold* sehingga predicted class label diperoleh dengan representasi *one hot*

Multilayer ANN

A.3. Aktivasi Neural Network dengan *Forward Propagation*

- Akan dideskripsikan proses forward propagation untuk menghitung output dari model MLP.
  - Semua unit pada hidden layer dihubungkan pada semua unit-unit layer input, sehingga unit aktivasi pada hidden layer $a_1^{(h)}$ adalah

$$z_1^{(h)} = a_0^{(in)} w_{0,1}^{(h)} + a_1^{(in)} w_{1,1}^{(h)} + \ldots + a_m^{(in)} w_{m,1}^{(h)}$$

$$a_1^{(h)} = \phi\left(z_1^{(h)}\right)$$

$z_1^{(h)}$: net input dan $\phi(\cdot)$: fungsi aktivasi

Multilayer ANN

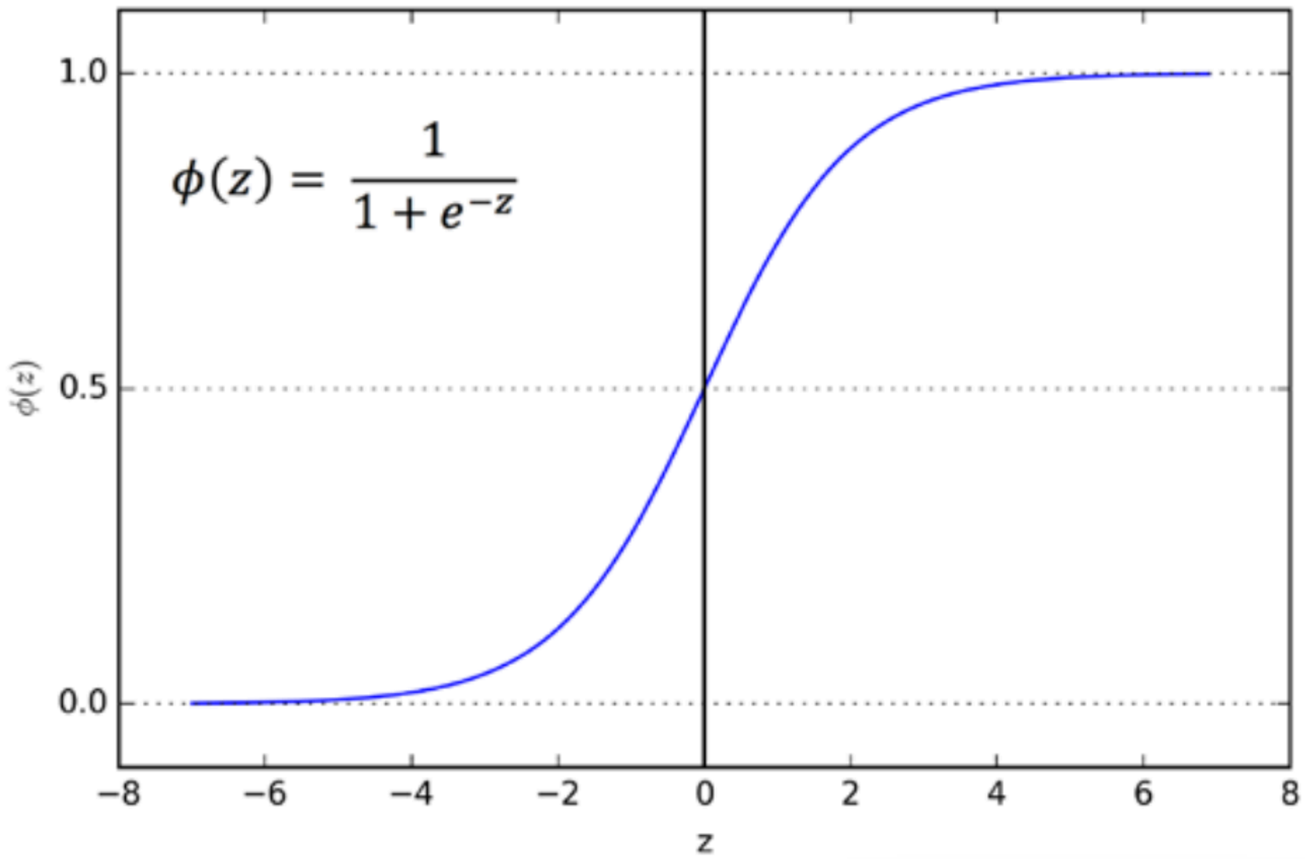
A.3. Aktivasi Neural Network dengan *Forward Propagation*

- Untuk masalah kompleks maka dibutuhkan fungsi aktivas non-linier pada MLP seperti fungsi sigmoid (logistik)

$$\phi(z) = \frac{1}{1 + e^{-z}}$$

- MLP adalah feedforward ANN (tiap layer berlaku sebagai input untuk layer berikutnya tanpa loop)

Fungsi Aktivasi Sigmoid (Logistik)



Meskipu namanya multilayer perceptron tetapi artificial neurons pada arsitektur ini adalah unit-unit sigmoid, bukan perceptron (lihat bab sebelumnya)

Multilayer ANN

A.3. Aktivasi Neural Network dengan *Forward Propagation*

- Representasi unit aktivasi dapat dibuat lebih padu dengan aljabar linier untuk satu sampel training $\mathbf{x}^{(i)}$

$$\mathbf{z}^{(h)} = \mathbf{a}^{(in)}\mathbf{W}^{(h)}$$
$$\mathbf{a}^{(h)} = \phi\left(\mathbf{z}^{(h)}\right)$$

$\mathbf{a}^{(in)}$: vektor fitur (dimensi $1 \times m$) dari sampel $\mathbf{x}^{(i)}$ plus sebuah unit bias

$\mathbf{W}^{(h)}$: matriks pembobot (dimensi $m \times d$), dan d adalah jumlah unit hidden layer

$\mathbf{z}^{(h)}$: vektor net input (dimensi $1 \times d$)

$\mathbf{a}^{(h)}$: vektor aktivasi $1 \times d$

Multilayer ANN

A.3. Aktivasi Neural Network dengan *Forward Propagation*

- Mengeneralisasi untuk semua n sampel training

$$\mathbf{Z}^{(h)} = \mathbf{A}^{(in)}\mathbf{W}^{(h)}$$
$$\mathbf{A}^{(h)} = \phi\left(\mathbf{Z}^{(h)}\right)$$

$\mathbf{A}^{(in)}$: matriks fitur (dimensi $n \times m$) dari semua sampel training $\mathbf{X}$

$\mathbf{W}^{(h)}$: matriks pembobot (dimensi $m \times d$), dan d adalah jumlah unit hidden layer

$\mathbf{Z}^{(h)}$: matriks net input (dimensi $n \times d$)

$\mathbf{A}^{(h)}$: matriks aktivasi $n \times d$

Multilayer ANN

A.3. Aktivasi Neural Network dengan *Forward Propagation*

- Selanjutnya, aktivasi layer output untuk keseluruhan sampel training

$$\mathbf{Z}^{(out)} = \mathbf{A}^{(h)}\mathbf{W}^{(out)}$$

$\mathbf{A}^{(h)}$: matriks fitur (dimensi $n \times d$)
$\mathbf{W}^{(out)}$: matriks pembobot (dimensi $d \times t$), dan t adalah jumlah unit output
$\mathbf{Z}^{(out)}$: matriks output (dimensi $n \times t$)

- Terakhir fungsi aktivasi sigmoid dihitung untuk memperoleh output berharga kontinyu

$$\mathbf{A}^{(out)} = \phi\left(\mathbf{Z}^{(out)}\right), \quad \mathbf{A}^{(out)} \in \mathbb{R}^{n \times t}$$

Multilayer ANN

B. Training sebuah ANN

Bagaimana melakukan learning (training) dari Multilayer ANN?

Multilayer ANN

B. Training sebuah ANN

- Akan dipelajari beberapa konsep dari **cost function logistik** dan algoritma **Backpropagation** untuk *learning* bobot-bobot.

Multilayer ANN

B.1. Menghitung Cost Function Logistik

- **Cost function logistik** sudah dipelajari sebelumnya (saat membahas regresi logistik)

$$J(\mathbf{w}) = -\sum_{i=1}^{n} y^{[i]} \log\left(a^{[i]}\right) + \left(1 - y^{[i]}\right) \log\left(1 - a^{[i]}\right), \text{ dan } a^{[i]} = \phi\left(z^{[i]}\right)$$

$a^{[i]}$: Aktivasi sigmoid dari sampel ke-i di dataset (i indeks sampel, bukan layer)

- Vektor target berdimensi $t \times 1$ dalam bentuk representasi one-hot encoding, sehingga aktivasi layer ke-3 dan target akan serupa dengan

$$a^{(out)} = \begin{bmatrix} 0{,}1 \\ 0{,}9 \\ \vdots \\ 0{,}3 \end{bmatrix}, \quad y = \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}$$

Multilayer ANN

B.1. Menghitung Cost Function Logistik

- Generalisasi cost function logistik untuk keseluruhan unit aktivasi yang berjumlah t

$$J(\mathbf{W}) = -\sum_{i=1}^{n}\sum_{j=1}^{t} y_j^{[i]} \log\left(a_j^{[i]}\right) + \left(1 - y_j^{[i]}\right) \log\left(1 - a_j^{[i]}\right)$$

- Untuk mengurangi overfitting ditambahkan regularisasi $L2$:

$$L2 = \frac{\lambda}{2}\sum_{l=1}^{L-1}\sum_{i=1}^{u_l}\sum_{j=1}^{u_{l+1}}\left(w_{j,l}^{(l)}\right)^2$$

u_l : menyatakan jumlah unit-unit dalam sebuah layer ke-l

Multilayer ANN

B.1. Menghitung Cost Function Logistik

- Cost function logistik untuk keseluruhan unit aktivasi yang berjumlah t dengan regularisasi tiap layer

$$J(\mathbf{W}) = -\left[\sum_{i=1}^{n}\sum_{j=1}^{t} y_j^{[i]} \log\left(a_j^{[i]}\right) + \left(1 - y_j^{[i]}\right) \log\left(1 - a_j^{[i]}\right)\right] + \frac{\lambda}{2}\sum_{l=1}^{L-1}\sum_{i=1}^{u_l}\sum_{j=1}^{u_{l+1}}\left(w_{j,i}^{(l)}\right)^2$$

- Tujuannya adalah untuk meminimumkan cost function $J(\mathbf{W})$ dengan cara menurunkan parameter $\mathbf{W}$ terhadap masing-masing bobot pada setiap layer:

$$\frac{\partial}{\partial w_{j,i}^{(l)}} J(\mathbf{w})$$

Bagian B.2 dan **B.3** akan menjelaskan algoritma **Backpropagation** untuk menghitung turunan parsial tersebut

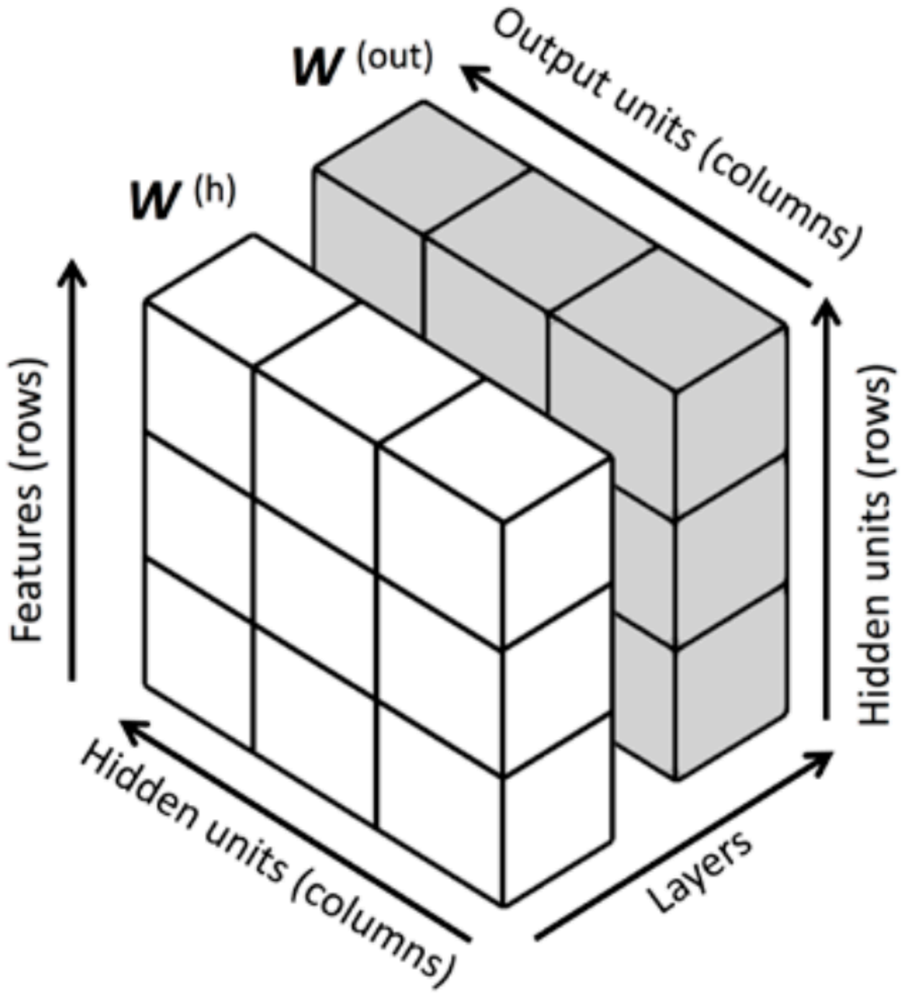
Multilayer ANN

B.1. Menghitung Cost Function Logistik

- Matriks $\mathbf{W}$ terdiri dari banyak matriks. Pada MLP matriks $\mathbf{W}$ terdiri dari $\mathbf{W}^{(h)}$ (menghubungkan input ke hidden layer) dan $\mathbf{W}^{(out)}$ (menghubungkan hidden layer ke output)

- $\mathbf{W}^{(h)}$ dan $\mathbf{W}^{(out)}$ jarang terjadi mempunyai jumlah baris dan kolom yang sama, meskipun gambar menunjukan hal tersebut.

Visualisasi Tensor 3D dari $\mathbf{W}$



Multilayer ANN

B.2. Intuisi untuk Algoritma Backpropagation

- **Backpropagation** adalah pendekatan komputasi yang efisien untuk menghitung turunan parsial cost function yang kompleks pada multilayer ANN
- Turunan tersebut digunakan untuk learning koefisien-koefisien pembobot sebagai parameter pada multilayer ANN tersebut
- Tantangan pada multilayer ANN adalah jumlah koefisien-koefisien pembobot bisa jadi sangat besar pada dimensi ruang fitur yang tinggi (berbeda dengan Adaline atau regresi logistik)
- Error surface pada cost function multilayer ANN ini **tidak convex** sebagai fungsi parameter (terdapat banyak minimum local yang harus ditanggulangi untuk menemukan minimum global)

Multilayer ANN

B.2. Intuisi untuk Algoritma Backpropagation

- Konsep **chain rule** digunakan untuk perhitungan turunan fungsi kompleks yang bersarang (*nested function*), seperti $f\left(g(x)\right)$

$$\frac{d}{dx}\left[f(g(x))\right] = \frac{df}{dg} \cdot \frac{dg}{dx}$$

- Misal kita punya 5 fungsi berbeda $f(x)$, $g(x)$, $h(x)$, $u(x)$, dan $v(x)$ dengan fungsi komposisi $F(x) = f(g(h(u(v(x)))))$, chain rule dapat digunakan untuk menghitung turunan ini:

$$\frac{dF}{dx} = \frac{d}{dx}F(x) = \frac{d}{dx}f(g(h(u(v(x))))) = \frac{df}{dg} \cdot \frac{dg}{dh} \cdot \frac{dh}{du} \cdot \frac{du}{dv} \cdot \frac{dv}{dx}$$

Multilayer ANN

B.2. Intuisi untuk Algoritma Backpropagation

- Teknik-teknik aljabar komputasi telah dikembangkan untuk memecahkan permasalahan di atas, disebut dengan **automatic differentiation**

- Automatic differentiation mempunyai dua mode, yaitu *forward* dan *reverse. Backpropagation* merupakan kasus khusus dari mode *reverse*
  - Aplikasi chain rule dengan mode forward (dari kiri ke kanan) bisa sangat mahal karena harus mengalikan matriks dengan matriks terus menerus, dan terakhir dengan sebuah vektor
  - Aplikasi chain rule dengan mode reverse (dari kanan ke kiri), mengalikan matriks dengan vektor akan menghasilkan vektor, yang kemudian dikalikan lagi dengan matriks, dst.

Perkalian matriks dan vektor pada backpropagation secara komputasi lebih murah dibandingkan dengan perkalian matriks dan matriks. Sehingga backpropagation menjadi salah satu algoritma popüler pada training neural networks

Multilayer ANN

B.3. Training Neural Networks via Backpropagation

1. Sebelum implementasi backpropagation, forward propagation diperlukan untuk menghasilkan aktivasi pada layer output:

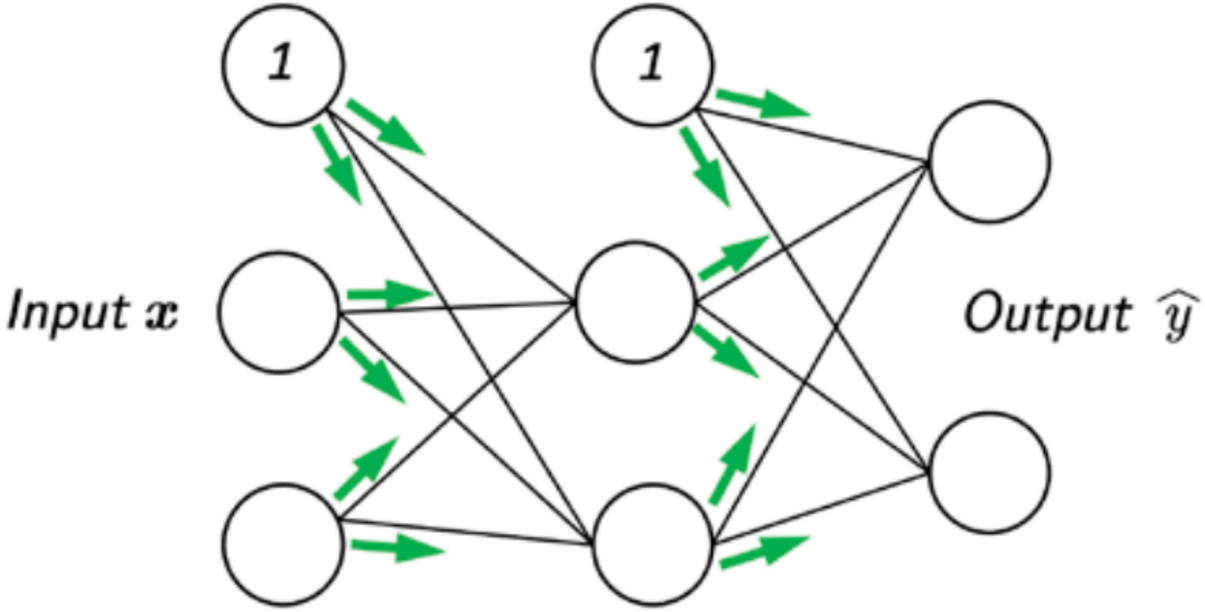
$$\mathbf{Z}^{(h)} = \mathbf{A}^{(in)}\mathbf{W}^{(h)}$$ (net input dari hidden layer)

$$\mathbf{A}^{(h)} = \phi\left(\mathbf{Z}^{(h)}\right)$$ (aktivasi dari hidden layer)

$$\mathbf{Z}^{(out)} = \mathbf{A}^{(h)}\mathbf{W}^{(out)}$$ (net input dari layer output)

$$\mathbf{A}^{(out)} = \phi\left(\mathbf{Z}^{(out)}\right)$$ (aktivasi dari layer output)

Ilustrasi Forward Propagate



Multilayer ANN

B.3. Training Neural Networks via Backpropagation

2. Error berpropagasi dari kanan ke kiri, mulai menghitung vektor error dari layer output:

$$\boldsymbol{\delta}^{(out)} = \mathbf{a}^{(out)} - \mathbf{y}$$

dimana $\mathbf{y}$ adalah vektor *true class label*

3. Menghitung error pada hidden layer

$$\boldsymbol{\delta}^{(h)} = \boldsymbol{\delta}^{(out)} \left(\mathbf{W}^{(out)}\right)^T \odot \frac{\partial \phi\left(z^{(h)}\right)}{\partial z^{(h)}},$$

$\frac{\partial \phi\left(z^{(h)}\right)}{\partial z^{(h)}} = \left(a^{(h)} \odot \left(1 - a^{(h)}\right)\right)$: turunan parsial dari fungsi aktivasi sigmoid,

Simbol $\odot$: perkalian antar elemen (*element-wise*)

Catatan: untuk bukti turunan parsial di atas silahkan lihat modul Chapter 2

Multilayer ANN

B.3. Training Neural Networks via Backpropagation

4. Hitung error matriks hidden layer $\delta^{(h)}$

$$\delta^{(h)} = \delta^{(out)} \left(\mathbf{W}^{(out)}\right)^T \odot \left(a^{(h)} \odot \left(1 - a^{(h)}\right)\right)$$

- $\mathbf{W}^{(out)}$: matriks berdimensi $h \times t$, maka $\left(\mathbf{W}^{(out)}\right)^T$ berdimensi $t \times h$
- $\delta^{(out)}$: matriks berdimensi $n \times t$
- $\delta^{(out)} \left(\mathbf{W}^{(out)}\right)^T$: matriks berdimensi $n \times h$
- $a^{(h)}$: matriks berdimensi $n \times h$

Maka hasil error matriks hidden layer $\delta^{(h)}$ adalah berdimensi $n \times h$

t : adalah jumlah label class output, h: adalah jumlah unit pada hidden layer,

n: adalah jumlah sampel training

Multilayer ANN

B.3. Training Neural Networks via Backpropagation

5. Setelah mendapatkan bagian δ, turunan dari cost function:

$$\frac{\partial}{\partial w_{i,j}^{(out)}} J(\mathbf{W}) = a_j^{(h)} \delta_i^{(out)}$$

$$\frac{\partial}{\partial w_{i,j}^{(h)}} J(\mathbf{W}) = a_j^{(in)} \delta_i^{(h)}$$

Turunan parsial lebih mudah diakumulasi dalam bentuk matriks:

$$\Delta^{(h)} = \left(\mathbf{A}^{(in)}\right)^T \delta^{(h)}$$

$$\Delta^{(out)} = \left(\mathbf{A}^{(h)}\right)^T \delta^{(out)}$$

Regularisasi untuk mengurangi overfitting bisa ditambahkan ke masing-masing layer l (hidden atau output):

$$\Delta^{(l)} := \Delta^{(l)} + \lambda^{(l)} \mathbf{W}^{(l)}$$

Multilayer ANN

B.3. Training Neural Networks via Backpropagation

6. Setelah gradien-gradien dihitung, kita dapat melakukan update dari pembobot-pembobot dimana mengambil langkah berlawanan arah dengan gradien untuk setiap layer l

$$\mathbf{W}^{(l)} := \mathbf{W}^{(l)} - \eta\Delta^{(l)}$$

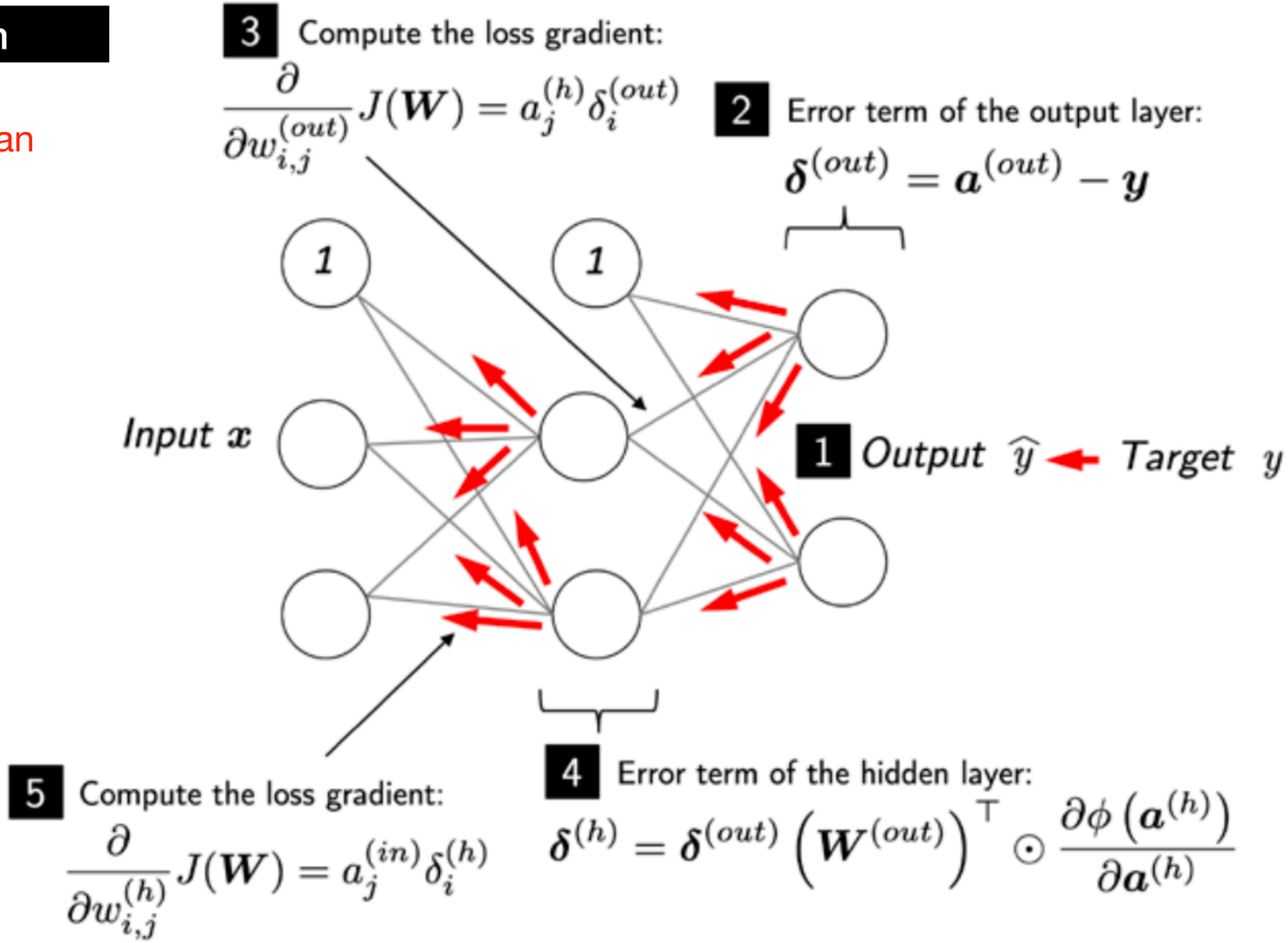
7. Ulangi dari langkah 1, sampai dianggap konvergen

Multilayer ANN

B.3. Training Neural Networks via Backpropagation

Ilustrasi algoritma backpropagation

Catatan: Nomor pada gambar ini tidak berpadanan dengan nomor langkah sebelumnya



Thank You